IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant :	Yuh-Cherng Wu	Art Unit :	2174
Serial No. :	10/829,145	Examiner :	Boris M. Pesin
Filed :	April 21, 2004	Conf. No. :	5096
Title :	SOFTWARE CONFIGURATION PROGRAM FOR SOFTWARE APPLICATIONS		

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

BRIEF ON APPEAL

(1) Real Party in Interest

SAP Aktiengesellschaft, the assignee of this application, is the real party in interest.

(2) Related Appeals and Interferences

There are no known related appeals and/or interferences.

(3) Status of Claims

Claims 1-15 are pending and stand rejected in the final office action dated June 4, 2009. Claims 1, 6, and 11 are independent. A notice of appeal is filed herewith.

Applicants appeal the rejections of all pending claims 1-15.

(4) Status of Amendments

All amendments have been entered. (No amendments were made after the Final Office Action). A listing of the current claims is provided in the Appendix provided with this Appeal Brief.

(5) Summary of Claimed Subject Matter

Independent claim 1 is directed to a method to be executed as part of a process for creating an executable configuration program. For example, the disclosure relates to "creating and executing a software configuration program for configuring software applications." (Spec.

1:5-7.) The software configuration can include "multiple steps that are successively executed." As an illustration, configuration of a complex software application may include the execution of a program (e.g., a "wizard" or "assistant") that includes multiple, sequential steps for properly configuring the application. (Spec. 2:13-16; 13:18 to 14:8; 14:13-15.) Each of the steps can be associated with user-selectable options. For example, in a first step of the configuration program a user may select from five options, and in a second step of the configuration program a user may select from three options. (Spec. 13:18 to 14:8.)

The method recites steps of "generating a user interface," "creating and storing in a repository [a] rule," and "binding the rule in the repository to [a] user-selectable option." Regarding the "generating a user interface step," claim 1 states that the generated user interface includes a "logic flow area for a user to define a command structure for the configuration program including at least one step." As an illustration, FIG. 4 of the specification shows a diagram of an example user interface for designing a configuration software application. (Spec. 5:21-22.) The user interface includes a logic flow area 84 that allows a user to arrange an ordering of steps to be presented to a user in the software configuration application. (Spec. 19:9-19; FIG. 4.) Claim 1 states that the user interface includes "a refinement area for the user to specify a configuration detail regarding a step arranged in the logic flow area." As an illustration, the graphical user interface shown in FIG. 4 includes a refinement area that can be activated for a particular step when a step is dragged from the flow chart palette 82 to the logic flow area 84. Once activated, the refinement area can allow a user to specify configuration details for a step, for example, explanatory text 92 that is displayed to a user of the executable configuration program. (Spec. 19:20 to 20:6; FIG. 4.) Claim 1 states that the user interface also includes "a rule palette for the user to create a rule, wherein the rule palette provides multiple conditional operators and entry fields." For example, the graphical user interface shown in FIG. 4 includes a rule palette 86 that allows a user to selected conditional operators (e.g., "If," "And," "Else," "Or," etc.) and entry fields (e.g., input areas where user selectable options can be specified) that can be arranged together to express rules that are executed by the software program during runtime. (Spec. 9:9-22; FIG. 4.)

Regarding the "creating and storing in a repository [a] rule step," claim 1 states recites a rule "that during execution of the configuration program determines which of the user-selectable options for the step are displayed when a specified user-selectable option of a previous step is selected." For example, a rule can determine, for user-selected options in a first graphical user interface of a configuration program, the user-selected options that will be displayed in a next graphical user interface of the configuration program. (Spec. 11:1-11.) The rules can be stored in a data repository 24. (Spec. FIG. 1; 16:22 to 17:6.) Claim 1 states that "the step and the previous step are arranged in the logic flow area and the user specifies the rule in the refinement area." As shown in example graphical user interface 24, steps can be arranged in the logic flow area and an ordering between them defined by the connectors 95. (Spec. 19:9-19; FIG. 4.) The rule can be defined in the refinement area. (Spec. 19:20 to 20:6; FIG. 4.)

Regarding the "binding the rule in the repository to [a] user-selectable option" step, claim 1 states that the rule is bound to the repository "so that during execution of the configuration program the rule is executed when the specified user-selectable option is selected." For example, when a user makes a selection in a first graphical user interface of the configuration program, a rule defined in the refinement area 88 in the user interface for creating the configuration program may be executed. (Spec. 16:7-10.)

Independent claim 6 is directed to a system. The system includes a "computer network," and a "service delivery device coupled to the network." Claim 6 states that "the service delivery device include[es] a processor and memory storing instructions that, in response to receiving a first type of request for access to a service, cause the processor to" perform certain operations. For example, a system 10 includes a processor 12 and random access memory 13, and is connected to a network 19 (*See* FIG. 1.) The operations will "create an executable program," "generated a user interface," "create and store in a repository a rule," and "bind the rule in the repository." Regarding the "create an executable configuration program" step, claim 6 states that a configuration program may be created "program that comprises multiple steps that are successively executed and wherein associated with each of the steps are user-selectable options." As an illustration, configuration of a complex software application may include the execution of a program (e.g., a "wizard" or an "assistant") that includes multiple, sequential steps for properly configuring the application. (Spec. 2:13-16; 13:18 to 14:8; 14:13-15.) Each of the steps can be associated with user-selectable options. For example, in a first step of the configuration program a user may select from five options, and in a second step of the configuration program a user may select from three options. (Spec. 13:18 to 14:8.)

Regarding the "generate a user interface step," claim 6 states that the generated user interface includes a "logic flow area for a user to define a command structure for the configuration program including at least one step." As an illustration, FIG. 4 of the specification shows a diagram of an example user interface for designing a configuration software application. (Spec. 5:21-22.) The user interface includes a logic flow area 84 that allows a user to arrange an ordering of steps to be presented to a user in the software configuration application. (Spec. 19:9-19; FIG. 4.) Claim 6 states that the user interface includes "a refinement area for the user to specify a configuration detail regarding a step arranged in the logic flow area." As an illustration, the graphical user interface shown in FIG. 4 includes a refinement area that can be activated for a particular step when a step is dragged from the flow chart palette 82 to the logic flow area 84. Once activated, the refinement area can allow a user to specify configuration details for a step, for example, explanatory text 92 that is displayed to a user of the executable configuration program. (Spec. 19:20 to 20:6; FIG. 4.) Claim 6 states that the user interface also includes "a rule palette for the user to create a rule, wherein the rule palette provides multiple conditional operators and entry fields." For example, the graphical user interface shown in FIG. 4 includes a rule palette 86 that allows a user to selected conditional operators (e.g., "If," "And," "Else," "Or," etc.) and entry fields (e.g., input areas where user selectable options can be specified) that can be arranged together to express rules that are executed by the software program during runtime. (Spec. 9:9-22; FIG. 4.)

Regarding the "create and store in a repository [a] rule step," claim 6 states recites a rule "that during execution of the configuration program determines which of the user-selectable options for the step are displayed when a specified user-selectable option of a previous step is selected." For example, a rule can determine, for user-selected options in a first graphical user interface of a configuration program, the user-selected options that will be displayed in a next graphical user interface of the configuration program. (Spec. 11:1-11.) The rules can be stored in a data repository 24. (Spec. FIG. 1; 16:22 to 17:6.) Claim 6 states that "the step and the previous step are arranged in the logic flow area and the user specifies the rule in the refinement area." As shown in example graphical user interface 24, steps can be arranged in the logic flow area and an ordering between them defined by the connectors 95. (Spec. 19:9-19; FIG. 4.) The rule can be defined in the refinement area. (Spec. 19:20 to 20:6; FIG. 4.)

Regarding the "bind the rule in the repository to [a] user-selectable option" step, claim 6 states that the rule is bound to the repository "so that during execution of the configuration program the rule is executed when the specified user-selectable option is selected." For example, when a user makes a selection in a first graphical user interface of the configuration program, a rule defined in the refinement area 88 in the user interface for creating the configuration program may be executed. (Spec. 16:7-10.)

Independent claim 11 is directed to a computer-readable storage device that can include instructions executed by a processor. For example, the system 10 can include non-volatile memory 14 that includes a design-time software program 16 used to generate a configuration software program and a run-time software program 18, as well as a processor 12. (*See* Spec. 7:11-17; FIG. 1.) Claim 11 states that when the instructions are executed by the processor, the processor will "create an executable configuration program," "generate a user interface," "create and store in a repository [a] rule," and "bind the rule in the repository to the specified user-selectable option."

Regarding the "create an executable configuration program" step, claim 11 states that a configuration program may be created "program that comprises multiple steps that are successively executed and wherein associated with each of the steps are user-selectable options." As an illustration, configuration of a complex software application may include the execution of a program (e.g., a "wizard" or an "assistant") that includes multiple, sequential steps for properly configuring the application. (Spec. 2:13-16; 13:18 to 14:8; 14:13-15.) Each of the steps can be associated with user-selectable options. For example, in a first step of the configuration program a user may select from five options, and in a second step of the configuration program a user may select from three options. (Spec. 13:18 to 14:8.)

Regarding the "generate a user interface step," claim 11 states that the generated user interface includes a "logic flow area for a user to define a command structure for the configuration program including at least one step." As an illustration, FIG. 4 of the specification shows a diagram of an example user interface for designing a configuration software application. (Spec. 5:21-22.) The user interface includes a logic flow area 84 that allows a user to arrange an ordering of steps to be presented to a user in the software configuration application. (Spec. 19:9-19; FIG. 4.) Claim 11 states that the user interface includes "a refinement area for the user to specify a configuration detail regarding a step arranged in the logic flow area." As an illustration, the graphical user interface shown in FIG. 4 includes a refinement area that can be activated for a particular step when a step is dragged from the flow chart palette 82 to the logic flow area 84. Once activated, the refinement area can allow a user to specify configuration details for a step, for example, explanatory text 92 that is displayed to a user of the executable configuration program. (Spec. 19:20 to 20:6; FIG. 4.) Claim 11 states that the user interface also includes "a rule palette for the user to create a rule, wherein the rule palette provides multiple conditional operators and entry fields." For example, the graphical user interface shown in FIG. 4 includes a rule palette 86 that allows a user to selected conditional operators (e.g., "If," "And," "Else," "Or," etc.) and entry fields (e.g., input areas where user selectable options can be specified) that can be arranged together to express rules that are executed by the software program during runtime. (Spec. 9:9-22; FIG. 4.)

Regarding the "create and store in a repository [a] rule step," claim 11 states recites a rule "that during execution of the configuration program determines which of the user-selectable options for the step are displayed when a specified user-selectable option of a previous step is selected." For example, a rule can determine, for user-selected options in a first graphical user interface of a configuration program, the user-selected options that will be displayed in a next graphical user interface of the configuration program. (Spec. 11:1-11.) The rules can be stored in a data repository 24. (Spec. FIG. 1; 16:22 to 17:6.) Claim 11 states that "the step and the previous step are arranged in the logic flow area and the user specifies the rule in the refinement area." As shown in example graphical user interface 24, steps can be arranged in the logic flow area and an ordering between them defined by the connectors 95. (Spec. 19:9-19; FIG. 4.) The rule can be defined in the refinement area. (Spec. 19:20 to 20:6; FIG. 4.)

Regarding the "bind the rule in the repository to [a] user-selectable option" step, claim 11 states that the rule is bound to the repository "so that during execution of the configuration program the rule is executed when the specified user-selectable option is selected." For example, when a user makes a selection in a first graphical user interface of the configuration program, a rule defined in the refinement area 88 in the user interface for creating the configuration program may be executed. (Spec. 16:7-10.)

(6) Grounds of Rejection to be Reviewed on Appeal

The Office Action (at page 2) rejected claim 1-3, 5-8, 10-13, and 15 under 35 U.S.C. § 103(a) as being unpatentable over SAP Wizard Builder in view of Kodosky et al. (U.S. Pat. No. 4,901,221). The Office Action (at page 15) rejected claims 4, 9, and 14 under 35 U.S.C. § 103(a) as being unpatentable over SAP Wizard Builder, in view of Kodosky, and further in view of Watson-Luke et al. (U.S. Pub. No. 2005/0114240).

Claims 1, 6, and 11 are independent. Applicants are appealing the rejections of all pending claims 1-15.

(7) Argument

For the following reasons, Applicants respectfully assert that the present claims are patentable over the references of record, and request that the above rejections be reversed.

I. Rejection of claims 1-15 — SAP Wizard Builder and Kodosky do not disclose at least the claim features highlighted in the discussion of claim 1 below.

By way of introduction, the claimed subject matter can involve a method that provides a graphical interface that allows a user to configure a software configuration program (e.g., a configuration program for creating a software application "wizard" or "assistant"). (Spec. 1:10-21.) Thus, the graphical interface can serve as a configuration utility for creating a configuration program. The software configuration program that is created can present to a user a series of screens, and on each of the screens, present several options to a user for selection. (Spec. 2:13-16; 13:18 to 14:8; 14:13-15.) Selection of an option in a screen can impact the options that are presented in a subsequently displayed screen. (Spec. 16:7-10.)

The graphical interface described in claims 1, 6 and 11includes "a logic flow area," a "refinement area," and a "rule palette." (*See, e.g.,* Spec. FIG. 4, reproduced herein.) A user of the graphical interface can arrange "steps" in the logic flow area to define an flow of screens that are presented to a user of the configuration program. (Spec. 19:9-19; FIG. 4.) Upon selection of a step in the logic flow area, a "refinement area" can be activated for the step. (Spec. 19:20 to 20:6; FIG. 4.) Once activated, the refinement area can allow a user to specify configuration details for a step, for example, explanatory text 92 that is displayed to a user of the executable configuration program. (*Id.*) The graphical user interface can also include a rule palette 86 that allows a user to selected conditional operators (e.g., "If," "And," "Else," "Or," etc.) and entry fields (e.g., input areas where user selectable options can be specified) that can be arranged together in the refinement area to express rules that are executed by the software program during runtime and in connection with a specific step. (Spec. 9:9-22; 21:13 to 22:12; FIG. 4.) After configuration using the graphical user interface (e.g., the interface displayed in FIG. 4), the software configuration program can be executed based on the rules and operations defined by the graphical user interface.

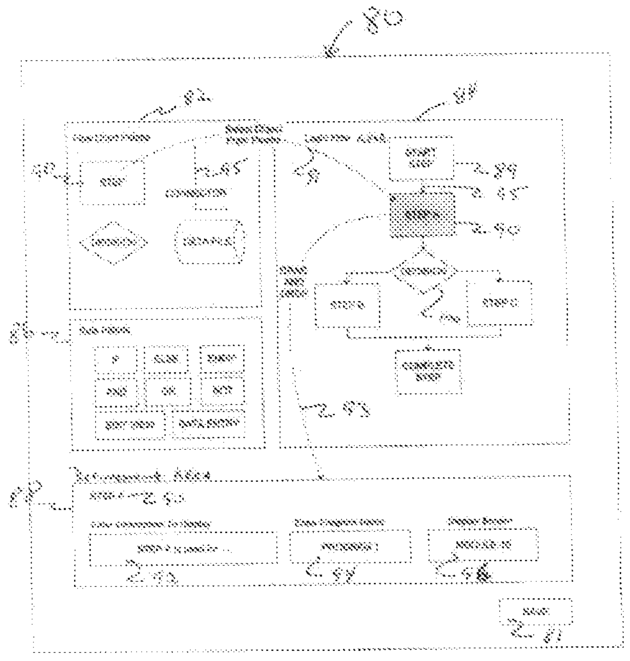


FIG. 4

Claim 1 recites that the user interface includes each of the three areas:

A method to be executed as part of a process for creating an executable configuration program that comprises multiple steps that are successively executed and wherein associated with each of the steps are user-selectable options, the method comprising:

generating a user interface including at least (i) a logic flow area for a user to define a command structure for the configuration program including at least one step, (ii) a refinement area for the user to specify a configuration detail regarding a step arranged in the logic flow area, and (iii) a rule palette for the user to create a rule, wherein the rule palette provides multiple conditional operators and entry fields;

creating and storing in a repository the rule that during execution of the configuration program determines which of the user-selectable options for the step are displayed when a specified user-selectable option of a previous step is selected, <u>wherein the step and the previous step are arranged in the logic flow area and the user specifies the rule in the refinement area</u>; and

binding the rule in the repository to the specified user-selectable option so that during execution of the configuration program the rule is executed when the specified user-selectable option is selected. (Emphasis added to language that the Office Action relies upon Kodosky as allegedly teaching.)

In contrast, SAP Wizard Builder and Kodosky, alone or in combination, fail to teach a user interface having the three areas recited in the present independent claims. For this reason, the references do not teach or suggest "***creating . . . the rule that during execution of the configuration program determines which of the user-selectable options for the step are displayed when a specified user-selectable option of a previous step is selected.***" The Office Action (at page 3) relies upon the SAP Wizard Builder as allegedly teaching the recited claim language. Specifically, the Office Action states that "rule created and binded (sic) to execute the rule binded (sic) to the specific option when user selects the user-selectable option as claimed above is shown in Figures 4-7 for example, also see annotation in figure below." (Office Action, page 3).

Applicant respectfully disagrees. Figures 4-7 of the SAP Wizard builder show a user interface for creating a wizard that includes identifying a name for the wizard (Figure 5), determining steps and creating descriptive texts for the steps (Figure 6), and assigning subscreens to the steps (Figure 7). The relied upon screenshots do not teach creating a rule that "determines which of the user-selectable options for [a] step are displayed when a specified user-selectable option of a previous step is selected." In some implementations, the recited subject matter includes creating a rule that governs which set of user-selectable options are displayed in a screen of a wizard, the displayed user-selectable options dependent upon what user-selectable options were selected in a previous screen. (Application, at page 16, lines 7-23.) SAP Wizard builder does not disclose a system for creating such a rule. Indeed, the Office Action has failed to articulate how the relied upon screenshots supposedly teach the recited language and Applicant submits that the screenshots do not teach the subject matter at issue.

Applicant notes that the Office Action references an "annotation in figure below," however, the Office Action did not include an annotated figure (only a non-annotated screenshot of Fig. 5 in the Office Action's discussion of claim 5). A claim rejection does not comply with 35 U.S.C. § 132 if it "is so uninformative that it prevents the applicant from recognizing and seeking to counter the grounds for rejection." *Chester v. Miller*, 906 F.2d 1574, 1578 (Fed. Cir. 1990).

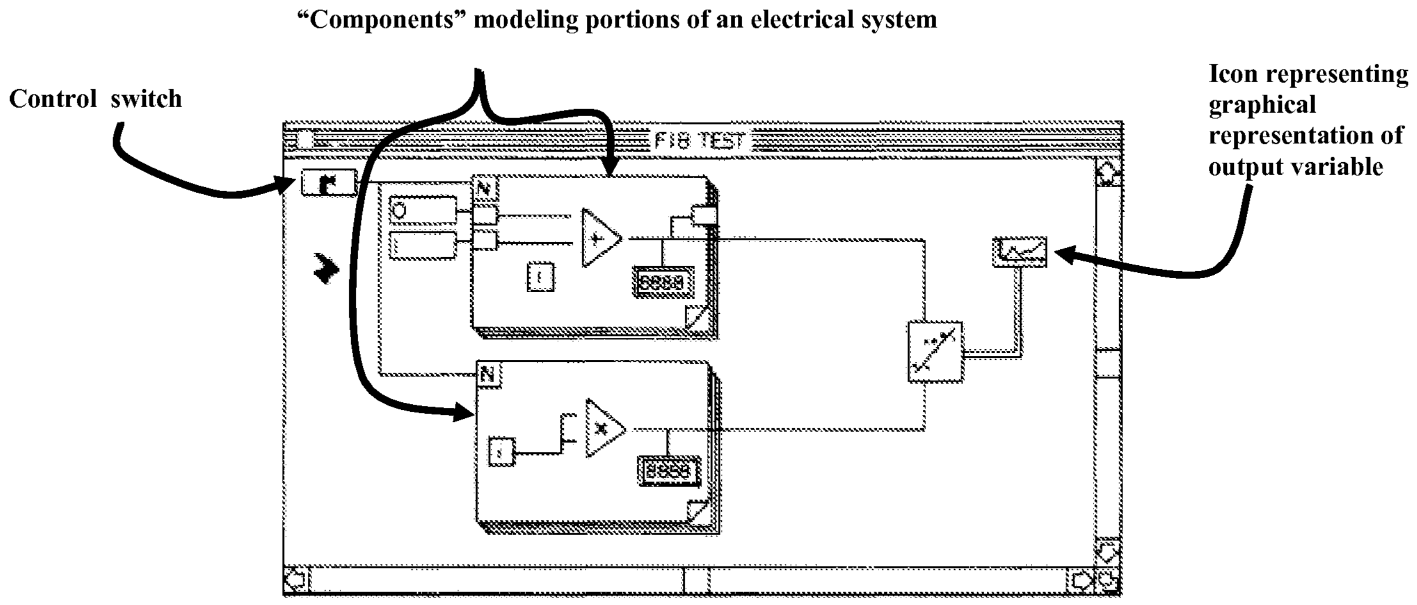
Kodosky does not provide the subject matter that SAP Wizard Builder is lacking. Kodosky describes a virtual instrument for modeling a manipulation of input variables to produce output variables, and does not teach "creating . . . the rule that during execution of the configuration program determines which of the user-selectable options for the step are displayed when a specified user-selectable option of a previous step is selected." (Kodosky, at col. 8, lines 3-23 and col. 4, lines 2-14.) Indeed, the Office Action did not rely upon Kodosky. Accordingly, the proposed combination fails to disclose the subject matter described in independent claim 1. Applicant respectfully submits that claim 1 is patentable over SAP Wizard Builder, Kodosky, and all other references in the record. Independent claims 6 and 11 contain similar language and are patentable for at least the same reasons. Dependent claims 2-5, 7-10, and 12-15 are patentable for at least the same reasons, and for the independently patentable features recited therein.

II. Rejection of claims 1-15 — The Office Action has failed to set forth an "articulated reason" to combine the SAP Wizard Builder and Kodosky references, and SAP Wizard Builder and Kodosky are non-analogous art.

The Office Action (at page 4) concedes that the features of the recited user interface are not taught by SAP Wizard builder. The Office Action then relies upon Kodosky as allegedly teaching these features (emphasized with underline in the above copy of claim 1).

Applicant respectfully disagrees. Kodosky describes a user interface that "permits a user to construct a virtual instrument." (Kodosky, col. 8, lines 3-23.) A user of Kodosky's system can virtually model a digital electrical system by wiring together "components" that represent the electrical components and adjusting "control dials and switches for providing variable input information." (Kodosky, at col. 8, lines 3-23 and 38-64; and col. 25, line 55 to col. 26, line 45.)

FIG. 57 of Kodosky (reproduced below) shows graphical representation of the output variables. (*Id.*)


"Components" modeling portions of an electrical system
Control switch
Icon representing graphical representation of output variable
FIB TEST

The Office Action states:

> It would have been obvious to one of ordinary skill in the art at the time of the invention to modify SAP and include a logic flow area, a refinement area, and a rule template area with the motivation to provide the user with a simpler method of creating the logic flow and to allow the user to more easily create a rule for execution.

Applicant respectfully disagrees. The Office Action improperly relies upon conclusory statements and impermissible hindsight to purport that the combination of SAP Wizard builder and Kodosky would have been obvious. Indeed, the Kodosky reference is non-analogous art, rendering it inappropriate as a reference under 35 U.S.C. § 103.

First, Applicant agrees with the Office Action that the recited graphical user interface of the present application can provide a simpler method of creating a logic flow. Applicant, however, disagrees with the Office Action's contention that modifying the SAP Wizard Builder to include a graphical interface similar to Kodosky's would have been obvious simply given a desire of one skilled in the art to provide a "simpler method of creating the logic flow." Indeed, other than this conclusory statement, the Office Action has failed to articulate a rationale for combining the SAP Wizard Builder with Kodosky. "[R]ejections on obviousness grounds cannot be sustained by mere conclusory statements; instead, there must be some ***articulated reasoning*** with some rational underpinning to support the legal conclusion of obviousness." *KSR Int'l Co. v. Teleflex Inc.*, 550 U.S. 398, 418 (2007) (emphasis added). Indeed, the Office Action has impermissibly relied upon hindsight to conclude that one skilled in the art seeking to provide a simpler method of creating a logic flow would look to the teachings of the Kodosky reference. *Id.* at 421 ("A factfinder should be aware, of course, of the distortion caused by hindsight bias and must be cautious of arguments reliant upon *ex post* reasoning.").

Kodosky describes a system that modifies a user-defined input variable with "components" and graphically displays the modified output signal. In contrast, the recited subject matter relates to a system that does not model the modification of an input signal in an electrical circuit. Rather, the recited subject matter relates to a system for creating an executable configuration program (e.g., a "wizard") for configuring software programs. (Application, at page 2, lines 11-16.) Such a system is not a "virtual instrument" and does not receive an input signal and modify the input signal to model an electrical instrument. The system can instead define a sequence of screens that will be displayed in the wizard. ***Kodosky is a non-analogous reference and should not be relied upon and a person of ordinary skill would not have combined it with the SAP Wizard***. Indeed, the Office Action has failed to "determine[e] the scope and content of the prior art" as required by *KSR* 550 U.S., at 399 (2007). A technique is obvious if "a person of ordinary skill in the art would recognize that it would improve ***similar*** devices in the same way." *KSR*, 550 U.S., at 401. In the present situation, however, the devices are not similar.

Accordingly, the proposed combination of SAP Wizard Builder and Kodosky is not a proper basis for the rejections. Applicant respectfully submits that claim 1 is patentable over SAP Wizard Builder, Kodosky, and all other references in the record. Independent claims 6 and 11 recite language that is similar to that in claim 1 and are patentable for at least the same reasons. Dependent claims 2-5, 7-10, and 12-15 are patentable for at least the same reasons as their independent claims, and for the independently patentable features therein.

III. Rejection of claims 4, 9, and 14

The Office Action (at page 15) rejected dependent claims 4, 9, and 14 under 35 U.S.C. § 103(a) as being unpatentable over SAP Wizard Builder, in view of Kodosky, and further in view of Watson-Luke et al. (U.S. Pub. No. 2005/0114240). Applicant respectfully submits that dependent claims 4, 9, and 14 are patentable for at least the same reasons as independent claims 1, 6, and 11, and for the additionally patentable features therein.

Please apply the required fee of $540 and any other charges or credits to Deposit Account No. 06-1050.

Respectfully submitted,

Date:October 19, 2009_______________

/j richard soderberg reg. no. 43,352/__________
J. Richard Soderberg
Reg. No. 43,352

Fish & Richardson P.C.
3200 RBC Plaza
60 South Sixth Street
Minneapolis, Minnesota 55402
Telephone: (612) 335-5070
Facsimile: (877) 769-7945

60601122.doc

Appendix of Claims

1. A method to be executed as part of a process for creating an executable configuration program that comprises multiple steps that are successively executed and wherein associated with each of the steps are user-selectable options, the method comprising:

generating a user interface including at least (i) a logic flow area for a user to define a command structure for the configuration program including at least one step, (ii) a refinement area for the user to specify a configuration detail regarding a step arranged in the logic flow area, and (iii) a rule palette for the user to create a rule, wherein the rule palette provides multiple conditional operators and entry fields;

creating and storing in a repository the rule that during execution of the configuration program determines which of the user-selectable options for the step are displayed when a specified user-selectable option of a previous step is selected, wherein the step and the previous step are arranged in the logic flow area and the user specifies the rule in the refinement area; and

binding the rule in the repository to the specified user-selectable option so that during execution of the configuration program the rule is executed when the specified user-selectable option is selected.

2. The method of claim 1 wherein the binding of the rule to the specified user-selectable option is performed by virtue of a designer selecting a user-selectable option for which to create the rule.

3. The method of claim 1 further comprising:

creating and storing in the repository a textual explanation of the step that describes available user-selectable options for the step; and

binding the textual explanation in the repository to the step so that during execution of the configuration program the textual explanation of the step is displayed.

4. The method of claim 3 wherein creating the textual explanation comprises translating the textual explanation into at least one different language.

5. The method of claim 1 further comprising evaluating the stability of a configured software application by executing the software application using a simulated user-selectable option.

6. A system comprising:

a computer network;

a service delivery device coupled to the network, the service delivery device including a processor and memory storing instructions that, in response to receiving a first type of request for access to a service, cause the processor to:

create an executable configuration program that comprises multiple steps that are successively executed and wherein associated with each of the steps are user-selectable options;

generate a user interface including at least (i) a logic flow area for a user to define a command structure for the configuration program including at least one step, (ii) a refinement area for the user to specify a configuration detail regarding a step arranged in the logic flow area, and (iii) a rule palette for the user to create a rule, wherein the rule palette provides multiple conditional operators and entry fields;

create and store in a repository the rule that during execution of the configuration program determines which of the user-selectable options for a step are displayed when a specified user-selectable option of a previous step is selected, wherein the step and the previous step are arranged in the logic flow area and the user specifies the rule in the refinement area; and

bind the rule in the repository to the specified user-selectable option so that during execution of the configuration program the rule is executed when the specified user-selectable option is selected.

7. The system of claim 6 wherein the memory stores instructions that, in response to receiving the first type of request, cause the processor to bind the rule to the specified user-selectable option by providing an ability to select a user-selectable option for which to create the rule.

8. The system of claim 6 wherein the memory stores instructions that, in response to receiving the first type of request, cause the processor to:

create and store in the repository a textual explanation of the step that describes available user-selectable options for the step; and

bind the textual explanation in the repository to the step so that during execution of the configuration program the textual explanation of the step is displayed.

9. The system of claim 8 wherein the memory stores instructions that, in response to receiving the first type of request, cause the processor to translate the textual explanation into at least one different language.

10. The system of claim 6 wherein the memory stores instructions that, in response to receiving the first type of request, cause the processor to evaluate the stability of a configured software application by executing the software application using a simulated user-selectable option.

11. A computer-readable storage device comprising instructions that, when executed by a processor, cause the processor to:

create an executable configuration program that comprises multiple steps that are successively executed and wherein associated with each of the steps are user-selectable options;

generate a user interface including at least (i) a logic flow area for a user to define a command structure for the configuration program including at least one step, (ii) a refinement area for the user to specify a configuration detail regarding a step arranged in the logic flow area, and (iii) a rule palette for the user to create a rule, wherein the rule palette provides multiple conditional operators and entry fields;

create and store in a repository the rule that during execution of the configuration program determines which of the user-selectable options for a step are displayed when a specified user-selectable option of a previous step is selected, wherein the step and the previous step are arranged in the logic flow area and the user specifies the rule in the refinement area; and bind the rule in the repository to the specified user-selectable option so that during execution of the configuration program the rule is executed when the specified user-selectable option is selected.

12. The storage device of claim 11 including instructions that, when executed by the processor, cause the processor to bind the rule to the specified user-selectable option by providing an ability to select a user-selectable option for which to create the rule.

13. The storage device of claim 11 including instructions that, when executed by the processor, cause the processor to:

create and store in the repository a textual explanation of the step that describes available user-selectable options for the step; and

bind the textual explanation in the repository to the step so that during execution of the configuration program the textual explanation of the step is displayed.

14. The storage device of claim 13 including instructions that, when executed by the processor, cause the processor to translate the textual explanation into at least one different language.

15. The storage device of claim 13 including instructions that, when executed by the processor, cause the processor to evaluate the stability of a configured software application by executing the software application using a simulated user-selectable option.

Evidence Appendix

None.

Related Proceedings Appendix

None.